CA377 Programming Fundamentals (Project)

Name	Declan Moore
Student Number	15375071
Programme	EC3
Module Code	CA377
Assignment Title	Programming Fundamentals (Project)
Submission date	8th December 2017
Module coordinator	Suzanne Little & Jennifer Foster

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged, and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the referencing guidelines found recommended in the assignment guidelines.

Name: Declan Moore Date: 05/12/2017

Introduction

Throughout my experience of the Enterprise Computing course the Programming Fundamentals module series has appeared on several occasions over the past two years. The most recent in this lecture series has been 'CA377: Programming Fundamentals (Project)'. Unlike previous modules in this series, which mostly dealt with weekly lectures followed by small exercises and a group project, this module allowed students to apply their accrued programming knowledge to a large-scale, semester long individual project. This project also concerned itself with programming within modern parameters such as cloud deployment and mobile UIs.

The app itself is called Park@DCU and its purpose is to provide parking information for any DCU campus to the app user, whether they are staff, students or visitors. This information can be presented in a variety of ways based on the queries of the user. It can allow Real Time Information of a specific carpark, provide historical data for a set date, provide the optimum carpark to use based on nearby facilities, or provide available carparks based on their opening hours. Users could also use the app to know what groups carparks are available to (staff, students or visitors), and they could differentiate queries based on whether they require disabled parking spaces or not.

System Architecture

Fundamentally, the main architecture of the Park@DCU app comprises of three main components. The first is an SQL database where we have stored information based on the campuses, carparks, facilities and historical parking data across the university. The following component of the app is a web service that returns real-time carp park occupancy data in JSON format. The final piece of our base Park@DCU architecture is our front-end. This is a Django-based web interface and it is designed to allow users query where they can park and occupancy of carparks at various times of the day, both historically and in real-time. Beyond these three components there are several other aspects to the project's development and design, but none provide the foundational function of the aforementioned items.

Django is an open-source web framework, written in Python that follows the model-view-template architectural pattern. We began the project by reacquainting ourselves with Git, Python and Django from modules of previous years. This involved use of a Linux terminal to install Django, create a Django project, a Django web application, creating a view, and mapping URLS before then pushing our new projects to our Gitlab repository. See figure 1 for description of MVT architecture and figure 2 for Gitlab workflow.

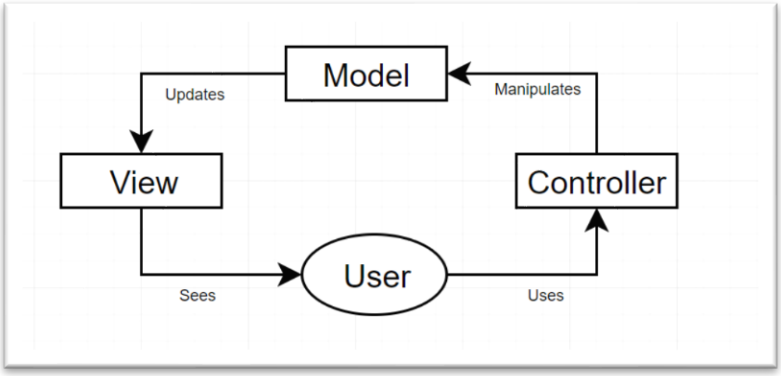

FIGURE 1

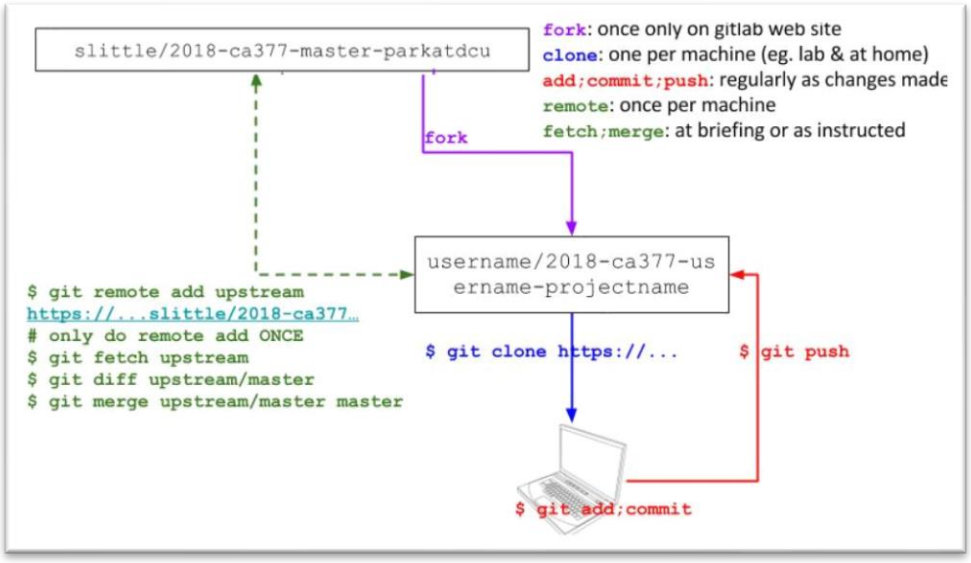

FIGURE 2

Once the setup for our Django and Git repository was completed it was time to create a database for our parking information. This was done by creating a MySQL database and importing provided .csv files from provided by fetching and merging with the master repository. Once the information had been imported to our MySQL database and the appropriate table structure had been designed we wrote several queries of that data to test it had been properly imported and structured for use in the app.

The final step in applying the foundations of the application was to integrate use of the webservice so that real-time information could be queried from it and displayed to the user. Here the project began using templates, an integral part to the Django framework. On the index html page of the app we had a form that would call the webservice 'on submit', it would then parse the JSON to extract real-time information about the desired carpark and present it to the user. Additionally, the app should also understand how to handle errors. This was achieved using 'if-else statements' and other simple python logic using the Django template language. Figure 3 displays an example of an early template for the webservice response.

FIGURE 3

Once the three steps above, we could then combine the three items and begin allowing a variety of queries for the application itself. We used the SQLite database engine to create the database for the Django application. The database schema was defined in the applications models.py file. We then populated the database from our previously exported MySQL database files into the application. These database structures were then visible from within the SQLite browser. Once imported we created other templates to allow the database information to be queried as well as the webservice from the previous stage. Figure 4 shows the information accessible through SQLite Browser after it's been integrated in the Django directories.

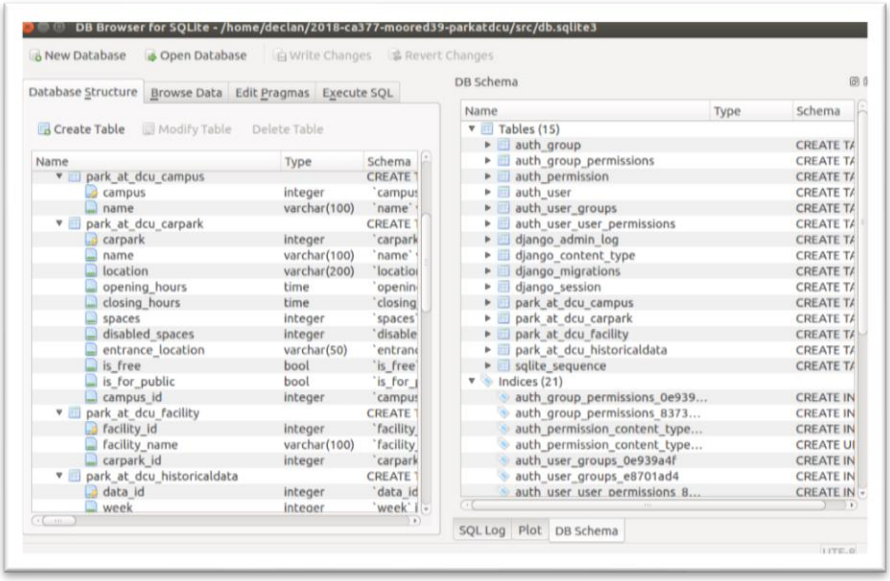

FIGURE 4

Once we could query the database and webservices in the various methods it was time to begin deploying the application. We employed use of PythonAnywhere.com for this due to its easy setup, in-browser development tools and standards to assist the deployment of Django projects. These factors meant that it would integrate well with the technologies already in use for the project and integration with the version control methods in place. We applied some text cases uses Django's tests.py methods to ensure our deployment applied correctly.

Once deployed to PythonAnywhere it was time to begin working on the front-end UI tools to create a better experience an accessibility for the user of the app. This involved designing the html inside our Django 'templates' directory to operate like a standard webpage or app under the DCU website and associated applications. We employed Django's static file settings to map the various files together for the deployment on PythonAnywhere. Using a combination of manually written CSS, Bootstrap templates, jQuery and JavaScript, we produced a responsive, appealing and functional front-end UI for the application that fell closely under DCU's already established branding guidelines and styling. Figure 5 shows a screenshot of the finished homepage while Figure 6 and Figure 7 show examples of CSS and JavaScript code used respectively.

As we can see, the architecture of this project can be appropriately represented through a step-by-step telling of the steps involved from Setup to Deployment.

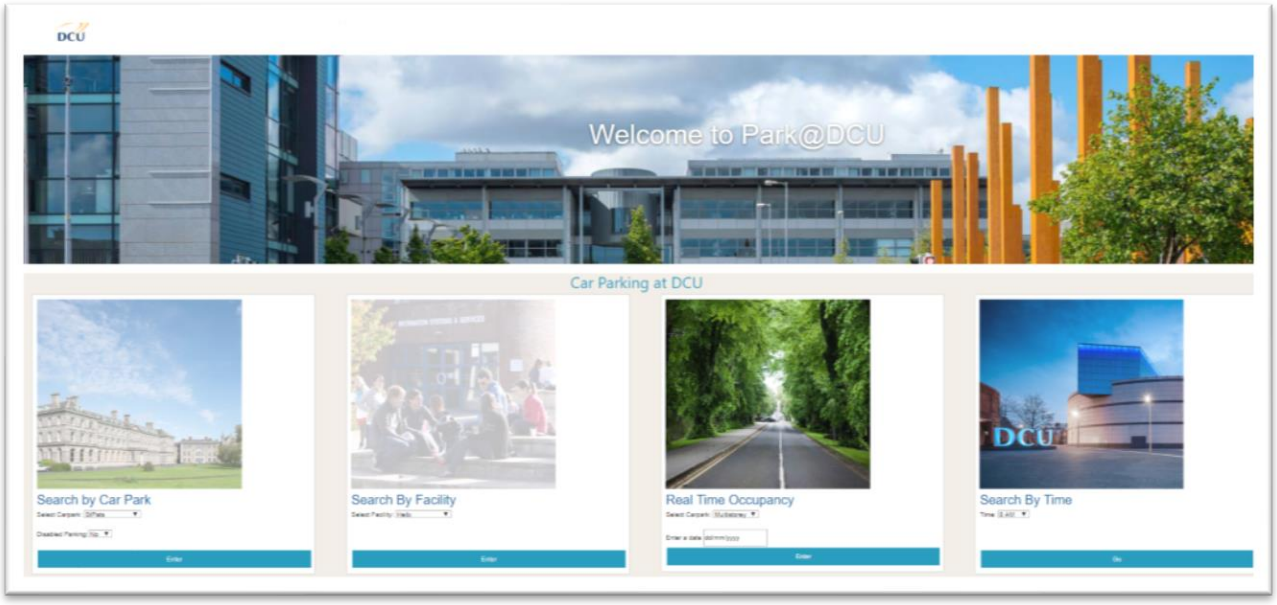

FIGURE 5

```
.btn {
  background-color: #259abb;
  color:#ffffff;
  text-decoration: none;
  border:none;
  display:block;
  margin:auto;
  width:10%;
  height:40px;
  text-align:center;
}

.btn:hover {
  background-color: #f1ede6;
  color:#259abb;
  text-decoration: none;
  border:none;
  display:block;
  margin:auto;
  width:10%;
  height:40px;
  text-align:center;

}



.learn-more {
    background-color:#f1ede6;
}
```

FIGURE 6

```
var myIndex = 0;
carousel1();

function carousel1() {
    var i;
    var x = document.getElementsByClassName("campus");
    for (i = 0; i < x.length; i++) {
       x[i].style.display = "none";
    }
    myIndex++;
    if (myIndex > x.length) {myIndex = 1}
    x[myIndex-1].style.display = "block";
    setTimeout(carousel1, 8500);
}
```

FIGURE 7

UI Description

For the user interface we started the design with five forms that our site visitors could use to query for information relating the carparks of DCU. Because of how specific these query forms were, we were able to combine two of them meaning only four forms were needed on the site.

I used elements from Bootstrap column templates to make all four columns appear on a page and take up the same amount of space. See Figure 8 for how this was achieved.

```
/* class for 4 columns */
ul.rig.columns-4 li {
    width: 22.5%; /* this value + 2.5 should = 25% */
}
```

FIGURE 8

Each query column was then turned into a box with an image inside it. The text and "Submit" button was then made the light blue colour used heavily on DCU's site. The button has a rollover effect when the cursor hovers over it. To make the page more stylistically appealing I wrote a carousel JavaScript for each of the column boxes to display a slideshow of images related to the queries. I applied a fading animation to this which is like some animations found on DCU's site.

The navigation bar at the top of the page clearly shows the DCU logo in the top left with a list of the navigation headings as found on the DCU site. The navigation bar's headings each then performs a dropdown to subitems and sub-subitems to some of those items then expand out to the right when hovered over. The navigation bar objects each correspond and map to an item on the DCU site's navigation bar. Although the navigation bar doesn't look or function identically to the DCU navigation bar I found it to be a better way of displaying information, particularly when there are that many items to display. See figure 9 for how the navigation bar works.

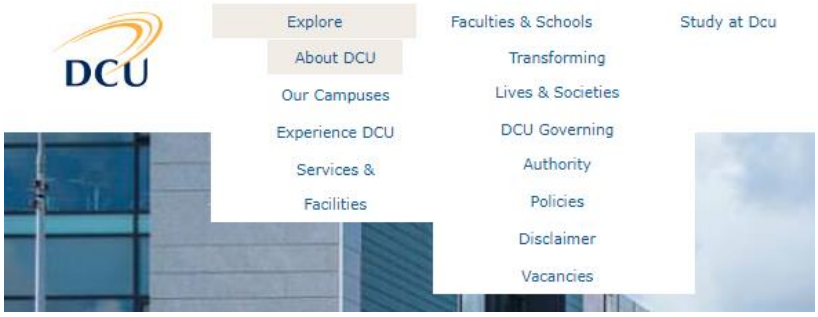

FIGURE 9

The banner image of the site uses the Bootstrap Jumbotron object to present a high-quality image of DCU's Glasnevin campus entrance. It also provides a welcome header which is also a hyperlink to the homepage and works from any page of the app. I decided to select the Jumbotron style because it allows me to choose an image that will be cropped or sized to suit the screen in use without stretching the image. Although this meant that at times the image might be more limited it always looks better than a squashed or stretched image.

The site's footer features similar items to those found on the DCU site again. Links to various Student and Staff services and apps. It also features a set of Social Network links in the center of the page and on the right a link to the Helix's website and the ESF website. Using CSS and an icon library for the Social links I was able to make the icons change to the platform primary colours when the cursor is hovered over them. See Figure 10 for an example of this.

FIGURE 10

Upon clicking any of the 'Submit' buttons beneath the queries on the Park@DCU app a corresponding page loads. The page is almost identical to the homepage except for the query results which appear between the jumbotron banner and the query columns. This decision to clone to homepage means users can easily try other queries without returning to the previous page. From my research I found that there are methods to apply similar interfaces more robustly using JavaScript, jQuery or Node.JS however with the time constraints of this project the static pages produced more than sufficient to provide a clear prototyping for the product. See Figure 11 for an example of an output table from the queries on the app.

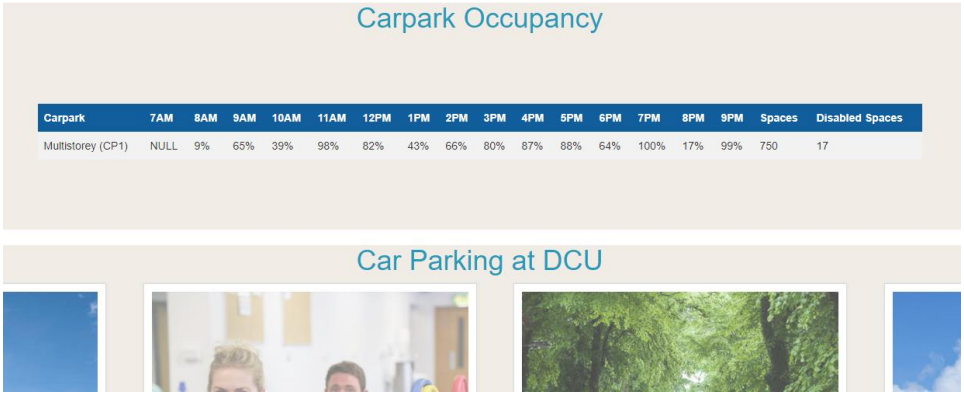

FIGURE 11

Using Source Code Repository (GitLab)

How did you use it?

For the project we used GitLab which is a Git repository hosting service. It allowed me to develop and design my application regardless of where I was working on it. Whether that be in a university lab following a briefing, on my laptop from the library or at home from my desktop PC. On both of my own devices I have Linux installed onto a VirtualBox virtual machine. Simply cloning or pulling the most recent repository to the machine I'm using, and I could pick up right where I left off.

Each week following briefings it was simply a case of fetching most recent updates from the lecturers' master repository and merging conflicts and then I'm ready to continue working on the app.

What worked/didn't work?

My development strategy worked without any problem until we reached the Deployment and UI stages. As far as deploying to PythonAnywhere I found there to be a few issues with setting that up properly and once it was established on that profile the additional steps to the version control flow meant sometimes forgetting to pull to PythonAnywhere or to Reload from the Web tab once there.

It also meant that despite static files working seamlessly locally or on an FTP/SCP client they proved very difficult without several hours of trial an error with various configurations and attempts to resolve problems.

I also began the bad habit of editing files directly from within the PythonAnywhere terminals and text editors which probably isn't best practice for version control as it results in mismatching of versions on Git and locally.

How could you have used it more effectively?

To use Git more effectively on future development projects I believe some behaviours need to be adopted from the beginning. For example, more concise and descriptive commit messages. I found myself throughout this project placing an arbitrary, unstructured and vague commit message on most of my commits. This lead me to difficulty finding a version later when looking to revert the project due to a bug issue.

For collaborative work I think that a better use of the user privileges for various other users might be a good idea to use. This can prevent a non-owner of the repository from being able to write to the master repository but still able to view the directories.

Committing more often is also a practice I would like to take with me to other repository based projects, to have a more varied set of versions to look back on in the case or problems arising later.

Additional Functionality

For my additional functionality I decided to tackle the site's UI to make the site follow the design styles set out by the existing DCU site. Some of these UI elements include the navigation menu, the image carousels, the additional information drop down content, and the footer.

The most challenging component I added was the image carousel above each of the queries. After initially creating the forms for each query, I felt there was something lacking from a visual point of view. I found some images on the Dublin City University Pixieset we were provided and made some edits and crops using Photoshop. Once I had four static images set for the queries it felt somewhat boring, I also felt that a single image didn't sufficiently convey the plurality of "facilities" or "campuses".

I looked at various methods of conveying these words which included a slideshow, an image collage or large icon sets. I felt that the two former would be too busy and the latter would probably involve designing the icons myself. Eventually I came across examples of image carousels on other sites and looked up guides on making them. I used a combination of HTML, CSS and JavaScript to write the carousel. The finished product essentially contains a series of images hidden using CSS. One image from the set is displayed upon opening the page, after a set interval the image begins to fade-out and the next image fades in. Initially with the four carousels all changing at the same time, it looked a bit strange, so I adjusted the timeout within the JavaScript functions slightly to offset this effect.

The Navigation menu contains dozens of items. It's based on the DCU site's own navigation menu. To contain that many items and still not encroach on the site's aesthetic, I made sub-item listings for each heading. Using CSS and hours of trial and error I managed to create a drop-down list for every option where the list of subheadings appear on cursor hover of the nav-bar items, then a second list of subitems appears to the right on hovering the cursor over the first set of subitems.

From the DCU site's Parking page there's a hefty amount of information pertaining the rules surrounding parking. I have added this information to my Park@DCU site so that users get a centralized grouping of all information relating to parking at DCU. Because of the volume of information, I have created dropdowns for each category of information using CSS. The categories include campus-specific rules for each campus as well as ticket prices for students, staff and visitors.

The footer of the site again representative of the footer on DCU's own site. The student and staff application and service links to the left, the social network image links in the center, and the Helix and ESF image links on the right. The social media images are taken from a Bootstrap library and I've used CSS to make them change colour on cursor hover, this is like the current slight image fade on DCU's version of this links. I felt like this design choice was a more modern update to what's currently in use.

Overall the attention to detail and extent of time I spent on this group of components of my Park@DCU app's UI are my submission for the additional functionality of the site. Having tested the site on both PC and mobile I feel like the time investment on perfecting the finished product has paid off.

Conclusion

What have you learn?

In conclusion, this module and project have combined many of the technologies and subjects previously covered in other modules studied in the Enterprise Computing course. I feel like I have a deeper understanding of how each component integrates together to form a cohesive application or project.

I've also managed to devise better personal tools for managing my time and deadlines. This is a byproduct of the regular briefings, deadlines and deliverables. Each week I had to find sufficient time to work on Park@DCU both inside out outside of the university.

Throughout this project at times we had to collaborate to accomplish larger tasks. Elements of the programming required to interact with the webservice were tasks we had limited experience with before. This meant that we were forced to collaborate in small groups and peer-educate each other on the subject. This is somewhat removed from other group work where often one person will do the heavy lifting and others might reuse their findings, in this case effective cooperation was required to achieve deadlines.

Overall, I really enjoyed CA377 and found it invaluable to implementing the technologies we have learned in Enterprise Computing.